Online Packet Scheduling with Bounded Delay and Lookahead*

Martin Böhm[1,2], Marek Chrobak[3], Łukasz Jeż[4], Fei Li[5], Jiří Sgall[1], and Pavel Veselý[1,6]

[1] Computer Science Institute of Charles University, Prague, Czech Republic, {sgall,vesely}@iuuk.mff.cuni.cz
[2] University of Bremen, Germany, martin.boehm@uni-bremen.de
[3] Department of Computer Science and Engineering, University of California, Riverside, USA, marek@cs.ucr.edu
[4] Institute of Computer Science, University of Wrocław, Poland, lje@cs.uni.wroc.pl
[5] Department of Computer Science, George Mason University, USA, lifei@cs.gmu.edu
[6] Department of Computer Science, University of Warwick, UK

Abstract

We study the *online bounded-delay packet scheduling problem (PacketScheduling)*, where packets of unit size arrive at a router over time and need to be transmitted over a network link. Each packet has two attributes: a non-negative weight and a deadline for its transmission. The objective is to maximize the total weight of the transmitted packets. This problem has been well studied in the literature; yet currently the best published upper bound is 1.828 [8], still quite far from the best lower bound of $\phi \approx 1.618$ [11, 2, 6].

In the variant of PacketScheduling with *s-bounded instances*, each packet can be scheduled in at most s consecutive slots, starting at its release time. The lower bound of ϕ applies even to the special case of 2-bounded instances, and a ϕ-competitive algorithm for 3-bounded instances was given in [5]. Improving that result, and addressing a question posed by Goldwasser [9], we present a ϕ-competitive algorithm for 4-*bounded* instances.

We also study a variant of PacketScheduling where an online algorithm has the additional power of *1-lookahead*, knowing at time t which packets will arrive at time $t+1$. For PacketScheduling with 1-lookahead restricted to 2-bounded instances, we present an online algorithm with competitive ratio $\frac{1}{2}(\sqrt{13}-1) \approx 1.303$ and we prove a nearly tight lower bound of $\frac{1}{4}(1+\sqrt{17}) \approx 1.281$. In fact, our lower bound result is more general: using only 2-bounded instances, for any integer $\ell \geq 0$ we prove a lower bound of $\frac{1}{2(\ell+1)}(1+\sqrt{5+8\ell+4\ell^2})$ for online algorithms with ℓ-lookahead, i.e., algorithms that at time t can see all packets arriving by time $t+\ell$. Finally, for non-restricted instances we show a lower bound of 1.25 for randomized algorithms with ℓ-lookahead, for any $\ell \geq 0$.

1 Introduction

Background. Optimizing the flow of packets across an IP network gives rise to a plethora of challenging algorithmic problems. In fact, even scheduling packet transmissions from a router across a specific network link can involve non-trivial tradeoffs. Several models for such tradeoffs

*M. Böhm, J. Sgall, and P. Veselý were supported by project 17-09142S of GA ČR and by the GAUK project 634217. M. Chrobak was supported by NSF grants CCF-1217314 and CCF-1536026. Ł. Jeż was supported by Polish National Science Center grant 2016/21/D/ST6/02402. F. Li was supported by NSF grant CCF-1216993. The conference version of this paper appeared in ISAAC 2016 [3]

have been formulated, depending on the architecture of the router, on characteristics of the packets, and on the objective function.

In the model that we study in this paper, each packet has two attributes: a non-negative weight and a deadline for its transmission. The time is assumed to be discrete (slotted), and only one packet can be sent in each slot. The objective is to maximize the total weight of the transmitted packets. We focus on the online setting, where at each time step the router needs to choose a pending packet for transmission, without the knowledge about future packet arrivals. This problem, which we call *online bounded-delay packet scheduling problem* (PacketScheduling), was introduced by Kesselman *et al.* [12] as a theoretical abstraction that captures the constraints and objectives of packet scheduling in networks that need to provide quality of service (QoS) guarantees. The combination of deadlines and weights is used to model packet priorities.

In the literature, the PacketScheduling problem is sometimes referred to as *bounded-delay buffer management in QoS switches*. It can also be formulated as the job-scheduling problem $1|p_j = 1, r_j| \sum w_j U_j$, where packets are represented by unit-length jobs with deadlines, with the objective to maximize the weighted throughput.

A router transmitting packets across a link needs to make scheduling decisions on the fly, based only on the currently available information. This motivates the study of online competitive algorithms for PacketScheduling. A simple online greedy algorithm that always schedules the heaviest pending packet is known to be 2-competitive [11, 12]. In a sequence of papers [7, 14, 8], this ratio was gradually improved, and currently the best published ratio is 1.828 [8]. The best lower bound, widely believed to be the optimal ratio, is $\phi = (1 + \sqrt{5})/2 \approx 1.618$ [11, 2, 6].

s-Bounded instances. In an attempt to bridge this gap, restricted models have been studied. In the *s-bounded* variant of PacketScheduling, each packet must be scheduled within k consecutive slots, starting at its release time, for some $k \leq s$ possibly depending on the packet; this k is then called the span of the packet. The lower bound of ϕ from [11, 2, 6] holds even in the 2-bounded case. A matching ϕ-competitive algorithm was given Kesselman *et al.* [12] for 2-bounded instances and by Chin *et al.* [5] for 3-bounded instances. Both results are based on the algorithm EDF_α, with $\alpha = \phi$, which always schedules the earliest-deadline packet whose weight is at least the weight of the heaviest pending packet divided by α (ties are broken in favor of heavier packets). EDF_ϕ is not ϕ-competitive for 4-bounded instances; however, a different choice of α yields a 1.732-competitive algorithm for the 4-bounded case [5].

Our contribution. We present a ϕ-competitive online algorithm for PacketScheduling restricted to 4-bounded instances, matching the lower bound of ϕ (see Section 3). This improves the results from [5] and answers the question posed by Goldwasser in his SIGACT News survey [9].

Our algorithm, called ToggleH, is based on a careful modification of EDF_ϕ. Unlike EDF_ϕ, under certain circumstances it schedules a packet lighter than the weight of the heaviest pending packet divided by ϕ. ToggleH is not memoryless, as it maintains one mark that may be assigned to a pending packet.

Algorithms with lookahead. In Sections 4 and 5, we investigate a variant of PacketScheduling where an online algorithm is able to learn at time t which packets will arrive by time $t + 1$. This property is known as *1-lookahead*. From a practical point of view, 1-lookahead corresponds to the situation in which a router can see the packets that are just arriving in the buffer and that will be available for transmission in the next time slot.

The notion of lookahead is quite natural and it has appeared in the online algorithm literature for paging [1], scheduling [15] and bin packing [10] since the 1990s. Ours is the first paper, to our knowledge, that considers lookahead in the context of packet scheduling.

Our contributions. We provide three results about PacketScheduling with lookahead. First, in Section 4, we present an online algorithm with 1-lookahead for 2-bounded instances with

competitive ratio of $\frac{1}{2}(\sqrt{13}-1) \approx 1.303$.

Then, in Section 5, we give a lower bound of $\frac{1}{4}(1+\sqrt{17}) \approx 1.281$ on the competitive ratio of algorithms with 1-lookahead which holds already for the 2-bounded case. Our argument is an extension of the lower bound proof of ϕ in [11, 2, 6]. In fact, our lower bound result is more general: using only 2-bounded instances, for any integer $\ell \geq 0$ we prove a lower bound of $\frac{1}{2(\ell+1)}(1+\sqrt{5+8\ell+4\ell^2})$ for online algorithms with ℓ-lookahead, that is algorithms that at time t can see all packets arriving by time $t+\ell$. It follows that there is no 1-competitive algorithm with any constant lookahead, even for 2-bounded instances.

Finally, in Section 6 we consider unrestricted instances, for which we generalize the lower bound of 1.25 by Chin and Fung [6], by proving that, for any constant $\ell \geq 0$, there is no better than 1.25-competitive randomized algorithm with ℓ-lookahead.

Subsequent work. There are two recent developments related to the presented work that need to be mentioned here. In paper [16], by a subset of authors of the present paper, a ϕ-competitive algorithm for general instances [16] was announced, thereby establishing the optimal competitive ratio of PacketScheduling (without lookahead). This new result notwithstanding, our ϕ-competitiveness proof for 4-bounded instances remains of independent interest, as it is significantly simpler than the proof in [16]. Further, unlike the algorithm in [16], our algorithm ToggleH uses very little memory (just one mark). Thus our proof may provide useful insights into the question whether there is a memory-efficient, or even memoryless, ϕ-competitive algorithm for PacketScheduling.

Also recently, Kobayashi [13] announced an optimal deterministic algorithm with 1-lookahead for 2-bounded instances, matching our lower bound of $\frac{1}{4}(1+\sqrt{17}) \approx 1.281$.

2 Definitions and Notation

Problem statement. Formally, we define the PacketScheduling problem as follows. The instance is a set of packets, with each packet p specified by a triple (r_p, d_p, w_p), where r_p and $d_p \geq r_p$ are integers representing the *release time* and *deadline* of p, and $w_p \geq 0$ is a real number representing the *weight* of p. Time is discrete, divided into unit *time slots*, also called *steps*. A *schedule* assigns time slots to some subset of packets such that (i) any packet p in this subset is assigned a slot in the interval $[r_p, d_p]$, and (ii) each slot is assigned to at most one packet. The objective is to compute a schedule that maximizes the total weight of the scheduled packets, also called the *profit*.

In the *s-bounded* variant of PacketScheduling, we assume that each packet p in the instance satisfies $d_p \leq r_p + s - 1$. In other words, this packet must be scheduled within some specified number, at most s, of consecutive slots starting at its release time.

Online algorithms. In the online variant of PacketScheduling, which is the focus of our work, at any time t only the packets released at times up to t are revealed. Thus an online algorithm needs to decide which packet to schedule at time t (if any) without any knowledge of packets released after time t.

As is common in the area of online optimization, we measure the performance of an online algorithm by its competitive ratio. An algorithm $\mathcal{A}$ is said to be *R-competitive* if, for all instances, the total weight of the optimal schedule (computed offline) is at most R times the weight of the schedule computed by $\mathcal{A}$.

We say that a packet p is *pending* for an algorithm at time t, if $r_p \leq t \leq d_p$ and p is not scheduled before time t. A (pending) packet p is *expiring* at time t if $d_p = t$, that is, it must be scheduled now or never. A packet p is *tight* if $r_p = d_p$; thus p is expiring already at its release time.

Algorithms with lookahead. In Sections 4 and 5, we investigate the PacketScheduling problem *with ℓ-lookahead*, mainly focusing on the case $\ell = 1$. With ℓ-lookahead, the problem definition changes so that at time t, an online algorithm can also see the packets that will be released at times $t+1, t+2, \ldots, t+\ell$, in addition to the pending packets. Naturally, only a pending packet can be scheduled at time t.

Other terminology and assumptions. We will make several assumptions about our problem that do not affect the generality of our results. First, we can assume that all packets have different weights. Any instance can be transformed into an instance with distinct weights through infinitesimal perturbation of the weights, without affecting the competitive ratio. Second, we assume that at each step there are at least two pending packets. (If not, we can always release some tight packets of weight 0 at each step.)

We define the *earliest-deadline relation* on packets, or *canonical ordering*, denoted $\prec$, where $x \prec y$ means that either $d_x < d_y$ or $d_x = d_y$ and $w_x > w_y$ (so the ties are broken in favor of heavier packets). At any step t, the algorithm maintains the earliest-deadline relation on the set of its pending packets. Throughout the paper, "earliest-deadline packet" means the earliest packet in the canonical ordering.

Regarding the adversary (optimal) schedule, we can assume that it satisfies the following *earliest-deadline property*: if packets p, p' are scheduled in steps t and t', respectively, where $r_{p'} \le t < t' \le d_p$ (that is, p and p' can be swapped in the schedule without violating their release times and deadlines), then $p \prec p'$. This can be rephrased in the following useful way: at any step, the optimum schedule transmits the earliest-deadline packet among all the pending packets that it transmits in the future.

3 An Algorithm for 4-bounded Instances

In this section, we present a ϕ-competitive algorithm for 4-bounded instances. Ratio ϕ is of course optimal as the lower bound instance is 2-bounded [11, 2, 6]. Up until now, the best competitive ratio for 4-bounded instances was $\sqrt{3} \approx 1.732$, achieved by algorithm $\mathrm{EDF}_{\sqrt{3}}$ in [5]. Our algorithm can be seen as a modification of EDF_ϕ. Unlike EDF_ϕ, in some circumstances our algorithm may schedule a packet lighter than w_h/ϕ, where h is the heaviest pending packet.

We remark that our algorithm uses memory; in particular, it marks one pending packet under certain conditions. This is in contrast to EDF_α, which is memoryless, i.e., it makes decisions solely based on the current set of pending packets. It is an interesting question whether there is a memoryless ϕ-competitive algorithm for 4-bounded instances.

Algorithm ToggleH. The algorithm maintains one mark that may be assigned to one of the pending packets. Initially, no packet is marked. For a given step t, we choose the following packets from among all pending packets:

h = the heaviest packet,

s = the second-heaviest packet,

f = the earliest-deadline packet with $w_f \ge w_h/\phi$, and

e = the earliest-deadline packet with $w_e \ge w_h/\phi^2$.

We then proceed as follows:

if h is not marked **or** $w_s \ge w_h/\phi$ **or** $d_e > t$
  schedule f
  if there is a marked packet **then** unmark it
  if $d_h = t+3$ **and** $d_f = t+2$ **then** mark h

else // h is marked and $w_s < w_h/\phi$ and $d_e = t$
schedule e
unmark h

We remark that all four packets h, s, e, and f are defined as we assume w.l.o.g. that there are at least two pending packets in each step. Note that when $f \neq h$, then the algorithm will always schedule f. This is because in this case f is a candidate for s, so the condition $w_s \geq w_h/\phi$ holds. The algorithm never specifically chooses s for scheduling – it is only used to determine if there is one more relatively heavy pending packet other than h. (But s *may* get scheduled if it so happens that $s = f$ or $s = e$.) Note also that, if $e \neq f$, then e is scheduled only in a very specific scenario, namely when all of the following conditions hold: e is expiring, h is marked, and $w_s < w_h/\phi$.

We distinguish two types of packets scheduled by Algorithm ToggleH: *f-packets*, scheduled using the first case, and *e-packets*, scheduled using the second case. Similarly, we refer to the steps as *f-steps* and *e-steps*.

Intuition. Let us give a high-level view of the analysis using charging schemes and an example that motivates both our algorithm and its analysis. The example consists of four packets j, k, f, h released in step 1, with deadlines $1, 2, 3, 4$ and weights $1-\varepsilon, 1-\varepsilon, 1, \phi$, respectively, for some small $\varepsilon > 0$. The optimum schedules all packets.

Compared to ToggleH, algorithm EDF_ϕ performs only f-steps; in our example it schedules f and h in steps 1 and 2, while j and k are lost. Thus the ratio is larger than ϕ. (In fact, after optimizing the threshold and the weight of h, this is the tight example for $\text{EDF}_{\sqrt{3}}$ on 4-bounded instances.) ToggleH deals with the predicament in step 2 of this example by marking h in step 1, and thus in step 2 it performs an e-step and schedules k which has the role of e and s in the algorithm.

This example and its variants are also important for our analysis. We analyze the algorithms by charging schemes, where the weight of each packet scheduled by the adversary is charged to one or more of the slots of the algorithm's schedule. If the weight charged to each slot is at most R times the weight of the packet scheduled by the algorithm in that slot, the algorithm is R-competitive. In the case of EDF_ϕ [5], one can charge the weight of each packet j scheduled by the adversary at time t either fully to the step where EDF_ϕ schedules j, if it is before t, or fully to step t otherwise. In our example, the weight charged to step 1 is $2-\varepsilon$ while EDF_ϕ schedules only weight 1, giving the ratio 2. Considering steps 1 and 2 together leads to a better ratio and after balancing the threshold it gives the tight analysis of $\text{EDF}_{\sqrt{3}}$ for 4-bounded instances.

Our analysis of ToggleH is driven by the variants of the example above where step 2 is an f-step. This may happen in several cases. One case is if in step 2 another packet s with $w_s \geq w_h/\phi$ arrives. If s is not scheduled in step 2, then s is pending in step 3, thus ToggleH schedules a relatively heavy packet in step 3, and we can charge a part of the weight of f, scheduled in step 3 by the adversary, to step 3. This motivates the definition of regular up and back charges below and corresponds to Case 5.1 in the analysis. Another case is when the weight of k is changed to $1/\phi - \varepsilon$. Then ToggleH performs an f-step because k is not a candidate for e, thus the role of e is taken by the non-expiring packet h. However, then the weight of the four packets charged to steps 1 and 2 in the way described above is at most ϕ times the weight of f and h; this corresponds to Case 5.2 of the analysis. Lemma 3.3 gives a subtle argument showing that in the 4-bounded case essentially these two variants of our example are the only difficult situations. Finally, in the original example, ToggleH schedules k in step 2 which is an e-step. Then again h is a pending heavy packet and we can charge some weight of f to step 3. Intuitively it is important that an e-step is performed only in a very specific situation where it is guaranteed that h can be scheduled in the next two steps (as it is marked) and that there is

no other packet of comparable weight due to the condition $w_s < w_h/\phi$. Still, there is a case to be handled: If more packets arrive in step 3, it is also possible that the adversary schedules h already in step 2 and we need to redistribute its weight. This case motivates the definition of the special up and back charges below.

Theorem 3.1. *Algorithm ToggleH is ϕ-competitive on 4-bounded instances.*

Proof. Fix some optimal adversary schedule. Without loss of generality, we can assume that this schedule satisfies the earliest-deadline property (see Section 2).

Let t be the current step. By h, f, e, and s we denote the packets from the definition of ToggleH. By j we denote the packet scheduled by the adversary. By h' and h'' we denote the heaviest pending packets in steps $t+1$ and $t+2$, respectively. We use the same convention for packets f, e, s, and j.

Our analysis uses a new charging scheme which we now define. The adversary packet j scheduled in step t is charged according to the first case below that applies:

1. If t is an e-step and $j = h$, we charge w_h/ϕ to step t and w_h/ϕ^2 to step $t-1$. We call these charges a *special up charge* and a *special back charge*, respectively. Note that the total charge is equal to $w_h = w_j$.

2. If j is pending for ToggleH in step t, charge w_j to step t. We call this charge a *full up charge*.

3. Otherwise j is scheduled before step t. We charge w_h/ϕ^2 to step t and $w_j - w_h/\phi^2$ to the step where ToggleH scheduled j. We call these charges a *regular up charge* and a *regular back charge*, respectively. We point out that the regular back charge may be negative, but this causes no problems in the proof.

We start with an easy observation that we use several times throughout the proof.

Lemma 3.2. *If an f-step t receives a regular back charge, then the up charge it receives is less than w_h/ϕ.*

Proof. For a regular up charge the lemma is trivial (with a slack of a factor of ϕ). For a full up charge, the existence of a regular back charge implies that the adversary schedules f after j, thus the earliest-deadline property of the adversary schedule implies that $j \prec f$, as both j and f are pending for the adversary at t. Thus ToggleH would schedule j if $w_j \geq w_h/\phi$. Finally, an f-step does not receive a special up charge. □

We examine packets scheduled by ToggleH in the order of time. For each time step t, if p is the packet scheduled at time t, we want to show that the charge to step t is at most ϕw_p. However, as it turns out, this will not always be true. In one case we will also consider the next step $t+1$ and the packet p' scheduled in step $t+1$, and show that the total charge to steps t and $t+1$ is at most $\phi(w_p + w_{p'})$.

Let t be the current step. We consider several cases.

Case 1: t is an e-step. By the definition of ToggleH, $w_e \geq w_h/\phi^2$ and $d_e = t$; the latter implies that step t receives no regular back charge. Note that the heaviest pending packet h' in step $t+1$ is unmarked at the beginning of step $t+1$ as only h is marked at the beginning of step t and we unmark it, while marking no other packet. This further implies that step $t+1$ is an f-step. Thus, step t receives no special back charge, which, combined with $d_e = t$, implies it receives no back charge of any kind.

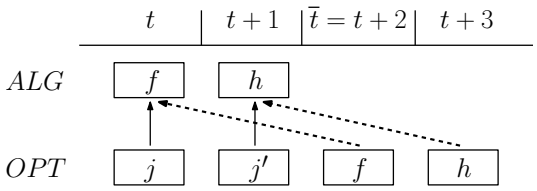

Figure 1: An illustration of the situation in Case 5.2. Up charges are denoted by solid arrows and back charges by dashed arrows.

Now we claim that the up charge is at most w_h/ϕ. For a special or regular up charge this follows from its definition. For a full up charge, packet j is pending at time t for ToggleH and $j \neq h$ (as for $j = h$ the special charges are used). This implies that $w_j < w_h/\phi$, as otherwise $w_s \geq w_h/\phi$ and t would be an f-step. Thus the full charge is $w_j \leq w_h/\phi$ as well.

Using $w_e \geq w_h/\phi^2$, the charge is at most $w_h/\phi \leq \phi w_e$ and we are done.

Case 2: t is an f-step and t does not receive a back charge. Then t can only receive an up-charge, and this up charge is at most $w_h \leq \phi w_f$, where the inequality follows from the definition of f.

Case 3: t is an f-step and t receives a special back charge. From the definition of special charges, the next step is an e-step, and therefore h' is marked at its beginning. Since the only packet that may be marked after an f-step is h, we thus have $h = h' = j'$, and the special back charge is w_h/ϕ^2. Note that $f \neq h$ as h is pending in step $t+1$. It follows that $f \prec h$ and as h is scheduled by the adversary in step $t + 1$, the adversary cannot schedule f after step t, so step t cannot receive a regular back charge.

We claim that the up charge to step t is at most w_f. Indeed, a regular up charge is at most $w_h/\phi^2 \leq w_f$, and a special up charge does not happen in an f-step. To show this bound for a full up charge, assume for contradiction that $w_j > w_f$. This implies that $j \neq f$ and, since ToggleH scheduled f, we have $d_j > d_f$. In particular j is pending at time $t + 1$. Thus $w_{s'} \geq w_j > w_f \geq w_h/\phi$, contradicting the fact that $t + 1$ is an e-step. Therefore the full charge is $w_j \leq w_f$, as claimed.

As $w_h \leq \phi w_f$, the total charge to t is at most $w_f + w_h/\phi^2 \leq w_f + w_f/\phi = \phi w_f$.

Case 4: t is an f-step, t receives a regular back charge and no special back charge, and $f = h$. The up charge is at most w_h/ϕ by Lemma 3.2 and the back charge is at most w_h, thus the total charge is at most $w_h + w_h/\phi = \phi w_h$, and we are done.

Case 5: t is an f-step, t receives a regular back charge and no special back charge, and $f \neq h$. Let $\bar{t}$ be the step when the adversary schedules f. We distinguish two sub-cases.

Case 5.1: In step $\bar{t}$, a packet of weight at least w_h/ϕ is pending for the algorithm. Then the regular back charge to t is at most $w_f - (w_h/\phi)/\phi^2 = w_f - w_h/\phi^3$. As the up charge to t is at most w_h/ϕ by Lemma 3.2, the total charge to t is at most $w_h/\phi + w_f - w_h/\phi^3 = w_f + w_h/\phi^2 \leq (1+1/\phi)w_f = \phi w_f$, and we are done.

Case 5.2: In step $\bar{t}$, no packet of weight at least w_h/ϕ is pending for the algorithm. In this case we consider the charges to steps t and $t + 1$ together. First, we claim the following.

Lemma 3.3. *ToggleH schedules h in step $t + 1$. Furthermore, step $t + 1$ receives no special charge and it receives an up charge of at most w_h/ϕ^2.*

Proof. Since $f \neq h$, we have $f \prec h$ and thus, using also the definition of $\bar{t}$ and 4-boundedness, $\bar{t} \leq d_f < d_h \leq t + 3$. The case condition implies that h is not pending at $\bar{t}$, thus ToggleH schedules h before $\bar{t}$. The only possibility is that ToggleH schedules h in step $t+1$, $\bar{t} = d_f = t+2$, and $d_h = t + 3$; see Figure 1 for an illustration. This also implies that ToggleH marks h in step t.

We claim that $w_{s'} < w_h/\phi$. Indeed, otherwise either s' is pending in step $t+2$, contradicting the condition of Case 5.2, or $d_{s'} = t+1 < d_h$, thus s' is a better candidate for f' than h, which contradicts the fact that the algorithm scheduled $f' = h$.

The claim also implies that $h' = h$, as otherwise $w_{s'} \geq w_h$. Since $h = h'$ is scheduled in step $t+1$, there is no marked packet in step $t+2$ and $t+2$ is an f-step; thus there is no special back charge to $t+1$.

We note that step $t+1$ is also an f-step, since ToggleH schedules h in step $t+1$ and $d_h > t+1$. Since $h' = h$ is marked when step $t+1$ starts and $w_{s'} < w_h/\phi$, the reason that step $t+1$ is an f-step must be that $d_{e'} > t+1$.

There is no special up charge to step $t+1$ as it is an f-step. If the up charge to step $t+1$ is a regular up charge, by definition it is at most $w_{h'}/\phi^2 = w_h/\phi^2$ and the lemma holds.

The only remaining case is that of a full up charge to step $t+1$ from a packet j' scheduled by the adversary in step $t+1$ and pending for ToggleH in step $t+1$. Since $j' \neq h$, it is a candidate for s', and thus $w_{j'} < w_h/\phi \leq w_f$. The earliest-deadline property of the adversary schedule implies that $j' \prec f$; together with $d_f = t+2$ and $w_{j'} < w_f$ this implies $d_{j'} = t+1$. Therefore $w_{j'} < w_h/\phi^2$, as otherwise j' is a candidate for e', but we have shown that $d_{e'} > t+1$. Thus the full up charge is $w_{j'} < w_h/\phi^2$ and the lemma holds also in the remaining case. □

By Lemma 3.3, step $t+1$ receives no special charge and an up charge of at most w_h/ϕ^2 and ToggleH schedules h in step $t+1$. Step $t+1$ thus also receives a regular back charge of at most w_h. So the total charge to step $t+1$ is at most $w_h/\phi^2 + w_h \leq w_f/\phi + w_h$. Moreover, using Lemma 3.2, the total charge to step t is at most $w_h/\phi + w_f$. Thus, the total charge to these two steps, where ToggleH schedules f and h, is at most $(w_h/\phi + w_f) + (w_f/\phi + w_h) = \phi(w_f + w_h)$.

In each case we have shown that a step or a pair of consecutive steps receive a total charge of at most ϕ times the weight of packets scheduled in these steps. Thus ToggleH is ϕ-competitive for the 4-bounded case. □

4 An Algorithm for 2-Bounded Instances with Lookahead

In this section, we present an algorithm for 2-*bounded* PacketScheduling *with 1-lookahead*, as defined in Section 2.

Consider some online algorithm $\mathcal{A}$. Recall that, for a time step t, packets *pending* for $\mathcal{A}$ are those that are released at or before time t and have neither expired nor been scheduled by $\mathcal{A}$ before time t. *Lookahead* packets at time t are the packets with release time $t+1$. For $\mathcal{A}$, we define the *plan* in step t to be the optimal schedule in the time interval $[t, \infty)$ that consists of pending and lookahead packets at time t and has the earliest-deadline property. For 2-bounded instances, this plan will only use slots t, $t+1$ and $t+2$. We will typically denote the packets in the plan scheduled in these slots by p_1, p_2, p_3, respectively. The earliest-deadline property then implies that if both p_1 and p_2 have release time t and deadline $t+1$ then p_1 is heavier than p_2. A similar condition holds for p_2 and p_3.

Algorithm CompareWithBias(α)**.** Fix some parameter $\alpha > 1$. At any time step t, the algorithm proceeds as follows:

let p_1, p_2, p_3 be the plan at time t
if $r_{p_2} = t$ **and** $w_{p_1} < \min(\, w_{p_2}\,,\, w_{p_3}\,,\, \frac{1}{2\alpha}(w_{p_2} + w_{p_3})\,)$
  then schedule p_2
else schedule p_1

Note that if the algorithm schedules p_2, then p_1 must be expiring, for otherwise $w_{p_1} > w_{p_2}$ (by canonical ordering). Also, the scheduled packet is at least as heavy as the heaviest expiring packet q, since clearly $w_{p_1} \geq w_q$ and the algorithm schedules p_2 only if $w_{p_1} < w_{p_2}$.

Theorem 4.1. *The algorithm* CompareWithBias(α) *is R-competitive for packet scheduling on 2-bounded instances for $R = \frac{1}{2}(\sqrt{13} - 1) \approx 1.303$ if $\alpha = \frac{1}{4}(\sqrt{13} + 3) \approx 1.651$.*

Let ALG be the schedule produced by CompareWithBias. Let us consider an optimal schedule OPT (a.k.a. schedule of the adversary) satisfying the canonical ordering, i.e., if a packet x is scheduled before a packet y in OPT then either y is released after x is scheduled or $x \prec y$. Recall that we are assuming w.l.o.g. that the weights of packets are different.

The analysis of CompareWithBias is based on a charging scheme. First we define a few packets by their schedule times, relative to some time t:

- e = packet scheduled in step $t-1$ in ALG,
- f = packet scheduled in step t in ALG,
- g = packet scheduled in step $t+1$ in ALG,
- h = packet scheduled in step $t+2$ in ALG,
- i = packet scheduled in step $t-1$ in OPT,
- j = packet scheduled in step t in OPT,
- k = packet scheduled in step $t+1$ in OPT,
- ℓ = packet scheduled in step $t+2$ in OPT.

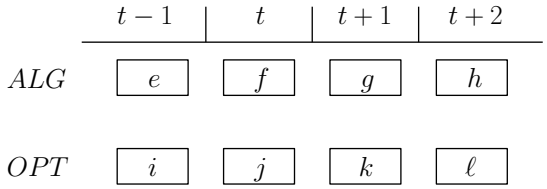

Figure 2: Packet definition.

Informal description of charging. We use three types of charges. The adversary's packet j in step t is charged using a *full charge* either to step $t-1$ if ALG schedules j in step $t-1$ or to step t if $w_f \geq w_j$ (including the case $f = j$) and f is not in step $t+1$ in OPT; the last condition assures that step t does not receive two full charges.

The second type are *split charges* that occur in step t if $w_f > w_j$, j is pending in step t in ALG and f is in step $t+1$ in OPT, i.e., step t receives a full back charge from f. In this case, we distribute the charge from j to f and another relatively large packet f' scheduled in step $t+1$ or $t+2$ in ALG; we shall prove that one of these steps satisfies $2\alpha \cdot w_j < w_f + w'_f$. We charge to step $t+2$ only when it is necessary, which allows us to prove that split-charge pairs are pairwise disjoint. Also, in this case we analyze the charges to both steps together, thus it is not necessary to fix a distribution of the weight to the two steps.

The remaining case is when $w_f < w_j$ and j is not scheduled in $t-1$ in ALG. We analyze these steps in maximal consecutive intervals, called *chains* and the corresponding charges are *chain charges*. Inside each chain we distribute the charge of each packet j scheduled at t in OPT to steps $t-1$, t and $t+1$, if these steps are also in the chain. The distribution of weights shall depend on a parameter δ. Packets at the beginning and at the end of the chain are charged in a way that minimizes the charge to steps outside of the chain. In particular, the step before a chain receives no charge from the chain.

Parameters and constants. We set the parameter α and constants δ and R which we will use in the analysis so that they satisfy the following equalities:

$$2 - \delta - \frac{R - 1 + 2\delta}{\alpha} = R \tag{1}$$

$$1 - 2\delta + 2\alpha\delta = R \tag{2}$$

$$1 + \frac{1}{2\alpha} = R \tag{3}$$

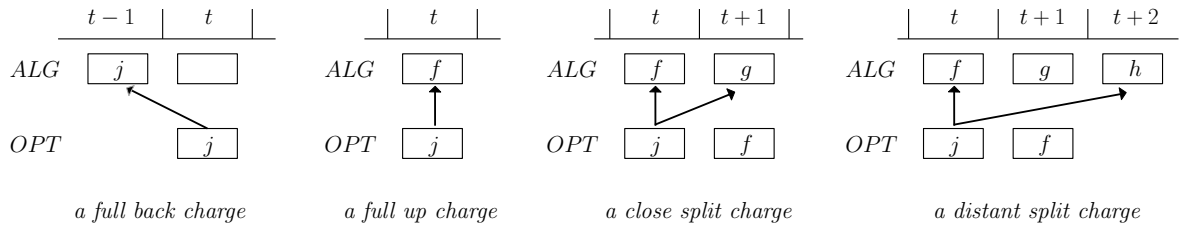

Figure 3: Non-chaining charges. Note that for split charges f is scheduled in step $t+1$ in OPT which follows from the fact that we do not charge j using a full up charge.

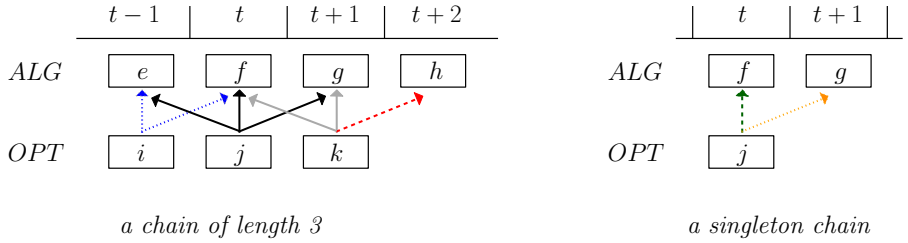

Figure 4: On the left, a chain of length 3 starting in step $t-1$ and ending in step $t+1$. The *chain beginning charges* are denoted by dotted (blue) lines, the *chain end charges* are denoted by gray lines and the *forward charge from a chain* is depicted by a dashed (red) arrow. Black arrows denote the *chain link charges*. On the right, an example of a singleton chain, with the *up charge from a singleton chain* denoted with a dashed (green) line and the *forward charge from a singleton chain* denoted with a dotted (orange) line.

By solving these equations we get $\alpha = \frac{1}{4}(\sqrt{13}+3) \approx 1.651$, $\delta = \frac{1}{6}(5-\sqrt{13}) \approx 0.232$, and $R = \frac{1}{2}(\sqrt{13}-1) \approx 1.303$. We will prove that the algorithm is R-competitive.

We also use the following properties of these constants:

$$2 - R - 3\delta = 0 \tag{4}$$

$$2 - R - 2\delta > 0 \tag{5}$$

$$1 - \delta - \frac{R - 1 + 2\delta}{2\alpha} > 0 \tag{6}$$

$$1 - \frac{R}{2\alpha} > 0 \tag{7}$$

$$3\alpha\delta < R \tag{8}$$

$$2 - \frac{R}{\alpha} < R \tag{9}$$

where (4) follows from (1) and (2).

Notations and the charging scheme. A step t for which $w_f < w_j$ and j is pending in step t in ALG is called *a chaining step*. A maximal sequence of successive chaining steps is called a *chain*. The chains with a single step are called *singleton chains*, the chains with at least two steps are called *long chains*.

The pair of steps that receives a split charge from the same packet is called a *split-charge pair*. The charging scheme does not specify the distribution of the weight to the two steps of the split-charge pair, as the charges to them are analyzed together.

Packet j scheduled in OPT at time t is charged according to the first rule below that applies. See Figures 3 and 4 for an illustration of different types of charges.

1. If j is scheduled in step $t-1$ in ALG (that is, $e=j$), charge w_j to step $t-1$. We call this charge a *full back charge*.

2. If $w_f \geq w_j$ and f is not scheduled in step $t+1$ in OPT (in particular, if $j=f$), charge w_j to step t. We call this charge a *full up charge*.

3. If $w_f > w_j$ and at least one of the following holds (in both cases, p_1 is the first packet in the plan at time t):
   - $2\alpha \cdot w_{p_1} < w_f + w_g$, or
   - g does not get a full back charge and $2\alpha \cdot (w_{p_1} - w_g) < w_f + w_g$,

   then charge w_j to the pair of steps t and $t+1$. We call this charge a *close split charge*.

4. If $w_f > w_j$, then charge w_j to the pair of steps t and $t+2$. We call this charge a *distant split charge*.

5. Otherwise step t is a chaining step, as $w_f < w_j$ and ALG does not schedule j in step $t-1$ by the previous cases. We distinguish the following subcases.

   (a) If step t is (the only step of) a singleton chain, then charge $\min(w_j, R \cdot w_f)$ to step t and $w_j - R \cdot w_f$ to step $t+1$ if $w_j > R \cdot w_f$. We call these charges an *up charge from a singleton chain* and a *forward charge from a singleton chain*.

   (b) If step t is the first step of a long chain, charge $2\delta \cdot w_j$ to step t, and $(1-2\delta) \cdot w_j$ to step $t+1$. We call these charges *chain beginning charges*.

   (c) If step t is the last step of a long chain, charge $\delta \cdot w_j$ to step $t-1$, $(R-1+2\delta) \cdot w_f$ to step t, and $(1-\delta) \cdot w_j - (R-1+2\delta) \cdot w_f$ to step $t+1$. We call these charges *chain end charges*; the charge to step $t+1$ is called a *forward charge from a chain*. (Note that we always have $(1-\delta) \cdot w_j > (R-1+2\delta) \cdot w_f$, since $w_j > w_f$ and $1-\delta = R-1+2\delta$ which follows from (4).)

   (d) Otherwise, i.e., if step t is inside a long chain, charge $\delta \cdot w_j$ to step $t-1$, $\delta \cdot w_j$ to step t, and $(1-2\delta) \cdot w_j$ to step $t+1$. We call these charges *chain link charges*.

To estimate the competitive ratio we need to show that each step or a pair of steps does not receive too much charge. We start with a useful observation about plans of Algorithm CompareWithBias(α), that will be used multiple times in our proofs.

Lemma 4.2. *Consider a time t, where the algorithm has two pending packets a, b and a lookahead packet c with the following properties: $d_a = t$, $(r_b, d_b) = (t, t+1)$, $(r_c, d_c) = (t+1, t+2)$, and $w_a < \min(w_b, w_c)$. If the algorithm schedules packet a in step t then the plan at time t is a, b, c, and $2\alpha \cdot w_a \geq w_b + w_c$.*

Proof. We claim that there is no pending or lookahead packet $q \notin \{b, c\}$ heavier than a. Suppose for a contradiction that such a q exists. Then a schedule containing packets q, b, c in some order is feasible and has larger profit than a, b, c. This implies that the plan does not contain a and thus a cannot be scheduled, contradicting the assumption of the lemma.

The schedule a, b, c is feasible and the claim above implies that it is optimal, thus it is the plan. It remains to show that $2\alpha \cdot w_a \geq w_b + w_c$, which follows easily by a contradiction: Otherwise $2\alpha \cdot w_a < w_b + w_c$ and CompareWithBias(α) would schedule b, contradicting the assumption of the lemma. □

Next, we will provide an analysis of full, split and chain charges, starting with full and split charges. We prove several lemmas from which the analysis follows. We fix some time slot t, and use the notation from Figure 2 for packets at time slots $t-1$, t, $t+1$ and $t+2$ in the schedule ALG of the algorithm and the optimal schedule OPT.

Analysis of full charges. Using Rules 1 and 2, if step t receives a full back charge, then the condition of Rule 2 guarantees that it will not receive a full up charge. This gives us the following observation.

Lemma 4.3. *Step t receives at most one full charge, i.e., a charge by Rule 1 or 2.*

Analysis of split charges. We now analyze close and distant split charges. The crucial property of split charges is that, similar to full charges, each step receives at most one split charge. Before we prove this, we establish several useful properties of split charges.

Lemma 4.4. *Let the plan at time t be p_1, p_2, p_3. If j is charged using a close or a distant split charge, then the following holds:*
(a) j is not scheduled by the algorithm in step $t-1$, i.e., j is pending for the algorithm in step t.
(b) $d_f = t+1$ and f is scheduled in step $t+1$ in OPT (that is, $k = f$). In particular, step t receives a full back charge.
(c) $d_j = t$ and $w_j \leq w_{p_1}$.
(d) $p_2 = f$.

Proof. By Rule 1, packet j would be charged using a full back charge if it were scheduled in step $t-1$, implying (a). As j is charged using a split charge, we have $w_f > w_j$. Thus, f is scheduled in step $t+1$ in OPT, since j is not charged using a full up charge, showing (b).

To show (c), note that if j is not expiring, then j and f would have equal deadlines. As we also have $w_f > w_j$, f would be scheduled before j in OPT by the canonical ordering, a contradiction. The inequality $w_j \leq w_{p_1}$ now follows from the definition of the plan.

It remains to prove (d). Towards contradiction, suppose that $f = p_1$. We know that j is expiring and thus it is not in the plan. If $d_{p_2} = t+1$ then the optimality of the plan implies $w_{p_2} > w_j$ (otherwise j, f, p_3 would be a better plan), so, since p_2 is not in OPT, we could improve OPT by scheduling f in step t and p_2 in step $t+1$.

Next, assume that $d_{p_2} = t+2$. The optimality of the plan implies that $w_{p_2} > w_j$ and $w_{p_3} > w_j$. Since both p_2, p_3 have deadline $t+2$, at least one of them is not scheduled in OPT. So OPT could be improved by scheduling f in step t and one of p_2 or p_3 in step $t+1$. In both cases we get a contradiction with the optimality of OPT. □

We show a useful lemma about a distant split charge from which we derive an upper bound on w_j, similar as the upper bound in the definition of close split charge.

Lemma 4.5. *If j is charged using a distant split charge, then $w_g < w_{p_3}$ where p_3 is the third packet in the plan at time t, and $d_g = t+1$.*

Proof. Suppose that $w_g \geq w_{p_3}$. Then, from Lemma 4.4(d) and the choice of $p_2 = f$ in the algorithm, we have that $2\alpha w_{p_1} < w_{p_2} + w_{p_3} \leq w_f + w_g$, so we would use the close split charge in step t, not the distant one. Thus $w_g < w_{p_3}$, as claimed.

To prove the second part, if we had $d_g = t+2$ then, since the algorithm chose g in step $t+1$ and also $d_{p_3} = t+2$, we would have $w_g \geq w_{p_3}$ – a contradiction. □

Lemma 4.6. *If j is charged using a distant split charge, then $2\alpha \cdot w_j < w_f + w_h$. (Recall that h is the packet scheduled in step $t+2$ in ALG.)*

Proof. Let p_1, p_2, p_3 be the plan in step t. By Lemma 4.4(d) we have that $f = p_2$. Thus $2\alpha \cdot w_{p_1} < w_{p_2} + w_{p_3}$ by the definition of the algorithm. By Lemma 4.4(c), j is expiring and $w_j \le w_{p_1}$. As $g \neq p_3$ by Lemma 4.5, the algorithm has p_3 pending in step $t+2$ where it is expiring, implying that $w_{p_3} \le w_h$. Putting it all together, we get $2\alpha w_j \le 2\alpha w_{p_1} < w_{p_2} + w_{p_3} \le w_f + w_h$. □

For a split charge from j in step t, let t' be the other step that receives the split charge from j; that is, $t' = t+1$ for a close split charge and $t' = t+2$ for a distant split charge. We now show that split-charge pairs are pairwise disjoint.

Lemma 4.7. *If j is charged using a split charge to a pair of steps t and t', then neither of t and t' is involved in another pair that receives a split charge from a packet $j' \neq j$.*

Proof. No matter which split charge we use for j, using Lemma 4.4(b), step $t+1$ does not receive a split charge from $k = f$. By a similar argument, since j is not scheduled in step $t-1$ in ALG, step t does not receive a close split charge from the packet scheduled in step $t-1$ in OPT.

It remains to prove that if j is charged using a distant split charge, then the packet ℓ scheduled in step $t+2$ in OPT is not charged using a split charge. (This also ensures that step t does not receive a distant split charge from a packet scheduled in step $t-2$ in OPT.)

For a contradiction, suppose that packet ℓ is charged using a split charge. Let p_1, p_2, p_3 be the plan in step t. Recall that g and h are the packets scheduled in steps $t+1$ and $t+2$ in ALG.

From Lemma 4.5, step $t+1$ does not receive a full back charge. Since we did not apply the close split charge for j in Rule 3, we must have

$$2\alpha(w_{p_1} - w_g) \ge w_f + w_g \ge w_f. \tag{10}$$

By Lemma 4.4(b) applied to step $t+2$, we get $d_h = t+3$. Since $d_{p_3} = t+2$, we get $w_{p_3} < w_h$. We now use Lemma 4.2 for step $t+1$ with $a = g, b = p_3$, and $c = h$. We note that all the assumptions of the lemma are satisfied: we have $d_g = t+1$, $(r_{p_3}, d_{p_3}) = (t+1, t+2)$, $(r_h, d_h) = (t+2, t+3)$, and $w_g < w_{p_3} < w_h$. This gives us that $2\alpha w_g \ge w_{p_3} + w_h > w_{p_3}$.

Since the algorithm schedules $f = p_2$ in step t, we have $2\alpha w_{p_1} < w_f + w_{p_3}$. Subtracting the inequality derived in the previous paragraph, we get $2\alpha(w_{p_1} - w_g) < (w_f + w_{p_3}) - w_{p_3} = w_f$ – a contradiction with (10). This completes the proof. □

The lemmas above allow us to estimate the total of full and split charges.

Lemma 4.8. *If j is charged using a split charge to a pair of steps t and t', then the total of full and split charges to steps t and t' does not exceed $R \cdot (w_f + w_{f'})$ where f' is the packet scheduled in step t' in ALG.*

Proof. Each of steps t and t' may receive a full charge, but each step at most one full charge from a packet of smaller or equal weight by Lemma 4.3 and charging rules.

First suppose that we use a distant split charge, or we use a close split charge and $2\alpha \cdot w_{p_1} < w_f + w_{f'}$. Then we have $2\alpha \cdot w_j < w_f + w_{f'}$ by Lemma 4.6 for a distant split charge or by $w_j \le w_{p_1}$ from Lemma 4.4(c) for a close split charge. Thus the total of full and split charges to steps t and t' is upper bounded by

$$w_f + w_{f'} + w_j < w_f + w_{f'} + \frac{w_f + w_{f'}}{2\alpha} = \left(1 + \frac{1}{2\alpha}\right) \cdot (w_f + w_{f'}) = R \cdot (w_f + w_{f'})$$

where we used (3) in the last step.

Otherwise, i.e., if we use a close split charge and $2\alpha \cdot w_{p_1} \ge w_f + w_{f'}$, then step $t' = t+1$ does not get a full back charge and we have $2\alpha \cdot (w_{p_1} - w_{f'}) < w_f + w_{f'}$ by Rule 3. By Lemma 4.4(c) we have $w_j \le w_{p_1}$ and $2\alpha \cdot (w_j - w_{f'}) < w_f + w_{f'}$. Also, step $t+1$ does not receive a full up

charge by Lemma 4.4(b). We thus bound the total of full and split charges to steps t and $t+1$ by

$$w_f + w_j < w_f + \frac{w_f + (2\alpha+1)\cdot w_{f'}}{2\alpha} = \left(1 + \frac{1}{2\alpha}\right)\cdot(w_f + w_{f'}) = R\cdot(w_f + w_{f'})$$

using (3) in the last step again. □

Analysis of chain charges. We now analyze chaining steps starting with a lemma below consisting of several useful observations. In particular, Part (c) motivates the name "chaining" for such steps.

Lemma 4.9. *If step t is a chaining step, then the following holds:*
(a) $d_j = t+1$,
(b) $d_f = t$.
Moreover, if step $t+1$ is also a chaining step, then
(c) j is scheduled by the algorithm in step $t+1$, i.e., $g = j$,
(d) $2\alpha\cdot w_f \geq w_j + w_k$ (recall that k is the packet scheduled in step $t+1$ in OPT).

Proof. Recall that Algorithm CompareWithBias(α) never schedules a packet lighter than the heaviest expiring packet. As in step t it schedules f with $w_f < w_j$ (by Rule 5 for chain charges) and j is pending (otherwise we use Rule 1), (a) follows. Furthermore, it follows that f is expiring in step t, because otherwise the algorithm would schedule j (or another packet of weight at least w_j), since both would have the same deadline and j is heavier. Thus (b) holds as well.

Now assume that step $t+1$ is also in the chain and for a contradiction suppose that $g \neq j$. Since j is expiring and pending for the algorithm in step $t+1$, we have $w_g > w_j$ and $w_k > w_g$ as step $t+1$ is in the chain.

Summarizing, the algorithm sees all packets f, j, g, k in step t (some are pending and some may be lookahead packets), and they are all distinct packets with $w_f < w_j < w_g < w_k$, $d_f = t$, $(r_j, d_j) = (t, t+1)$, and both g and k can be feasibly scheduled at time $t+1$. Thus, independently of the release times and deadlines of g and k, the plan at time t containing f would not be optimal – a contradiction. This proves that (c) holds.

Finally, we show (d). Since f is expiring in step t by (b) and both j and k are considered for the plan at time t and satisfy $(r_j, d_j) = (t, t+1)$, $(r_k, d_k) = (t+1, t+2)$ by (a), $w_f < w_j < w_k$, we use Lemma 4.2 with $a = f, b = j$, and $c = k$ and get the inequality in (d). □

First we show that chaining steps does not receive charges of other types.

Lemma 4.10. *If step t is a chanining step, then t does not receive a full charge or a split charge.*

Proof. By Lemma 4.9, f is expiring, thus step t does not receive a full back charge. As $w_j > w_f$, the step also does not get a full up charge or a split charge from step t. So it remains to show that f does not receive a split charge.

First observe that step t cannot receive a close split charge from step $t-1$ in OPT, because j is pending in step t in ALG, while Lemma 4.4(b) states that a split charge from step $t-1$ would require j to be scheduled at time $t-1$ in ALG.

Finally, we show that step t does not receive a distant split charge. For a contradiction, suppose that step t receives a distant split charge from the packet x scheduled in step $t-2$ in OPT. Let p_1, p_2, p_3 be the plan in step $t-2$. According to Lemma 4.4(d) and (b), p_2 is scheduled in step $t-2$ in ALG and in step $t-1$ in OPT. Moreover, by Lemma 4.4(c), x is pending and expiring in step $t-2$ and $w_{p_1} \geq w_x$. As the algorithm scheduled p_2 in step $t-2$ we get $r_{p_2} = t-2$ and $w_{p_1} < w_{p_3}$.

Observe that p_3 is not scheduled in OPT, since it is expiring in step t and j is not expiring, by Lemma 4.9(a). Thus we could increase the weight of OPT if we scheduled p_2 in step $t-2$ instead of x and p_3 in step $t-1$. This contradicts the optimality of OPT. □

We now analyze how much charge does each chaining step get.

Lemma 4.11. *If step t is a chaining step, then it receives a charge of at most $R \cdot w_f$.*

Proof. By Lemma 4.10, step t does not receive any full or split charges; therefore we just need to prove that the total of chain charges to step t does not exceed $R \cdot w_f$.

Case 1: t is the last step of a chain. If t is the only step in the chain then Rule 5a implies directly that the charge to t is at most $R \cdot w_f$. Otherwise, Lemma 4.9(c) implies that f is scheduled in step $t-1$ in OPT, and thus the charge from step $t-1$ is $(1-2\delta) \cdot w_f$. The charge from step t is $(R-1+2\delta) \cdot w_f$ by Rule 5c. So the total charge is at most $R \cdot w_f$.

Case 2: t is not the last step of a chain. Since step $t+1$ is also in the chain, by Lemma 4.9(c) we have that j is scheduled in step $t+1$ in ALG and OPT has a packet k with $w_k > w_j$ in step $t+1$. From Lemma 4.9(d) we know that $2\alpha \cdot w_f \geq w_j + w_k$.

There are two sub-cases. If t is the first step of the chain, then the charge to t is at most

$$2\delta \cdot w_j + \delta \cdot w_k \leq \tfrac{3}{2}\delta \cdot (w_j + w_k) \leq 3\alpha\delta \cdot w_f < R \cdot w_f,$$

where the last inequality follows from (8). Otherwise, using Lemma 4.9(c), f is scheduled in step $t-1$ in OPT, so the total charge to step t is at most

$$(1-2\delta) \cdot w_f + \delta \cdot w_j + \delta \cdot w_k \leq (1-2\delta) \cdot w_f + 2\alpha\delta \cdot w_f = R \cdot w_f$$

where the last equality follows from (2). □

Analysis of forward charges from chains. We now show that a forward charge from a chain does not cause an overload on the step just after the chain which may also get both a full charge and a split charge. (This is the only case when a step receives charges of all three types.)

For the following lemmas we assume that step $t-1$ is a chaining step. Recall that i is the packet scheduled in step $t-1$ in OPT and e is the packet scheduled in step $t-1$ in ALG. First, we prove some useful observations.

Lemma 4.12. *If step t receives a forward charge from a chain, then the following holds*
(a) $j \neq e$ (that is, j is not charged using a full back charge),
(b) $w_f \geq w_j$,
(c) $w_f \geq w_i$.
Moreover, if step t is not in a split-charge pair:
(d) j is charged using a full up charge to step t,
(e) step t does not receive a full back charge.

Proof. Part (a) holds because e is expiring in step $t-1$, by Lemma 4.9(b). Part (b) follows from (a) and the fact that step t is not chaining.

To show (c), Lemma 4.9(a) implies that $d_i = t$. Also, since $i \neq e$, i is pending in ALG in step t. Now (c) follows, because each packet scheduled by the algorithm is at least as heavy as the the heaviest expiring packet.

Part (d) follows from (a) and (b) and the assumption that t is not in a split-charge pair. Part (e) follows from (d) and Lemma 4.3. □

Note that f may be the same packet as i or j. We start with the case in which f is not in a split-charge pair.

Lemma 4.13. *If step t receives a forward charge from a chain C and t is not in a split-charge pair, then the total charge to step t is at most $R \cdot w_f$.*

Proof. The proof is by case analysis, depending on the relative weights of j and e, and on whether C is a singleton or a long chain. In all cases we use Lemma 4.12 and the charging rules to show upper bounds on the total charge.

Case 1: $w_j < w_e$.

Case 1.1: The chain C is long. The charge to step t is then at most

$$\begin{aligned} w_j + (1-\delta)\cdot w_i - (R-1+2\delta)\cdot w_e &< w_j + (1-\delta)\cdot w_i - (R-1+2\delta)\cdot w_j \\ &= (2-R-2\delta)\cdot w_j + (1-\delta)\cdot w_i \\ &\le (2-R-2\delta)\cdot w_f + (1-\delta)\cdot w_f \qquad (11) \\ &= (3-R-3\delta)\cdot w_f = w_f\,. \qquad (12) \end{aligned}$$

To justify inequality (11), note that $2-R-2\delta \ge 0$ by (5) and $1-\delta \ge 0$ by the choice of δ, so we can apply inequalities $w_j \le w_f$ and $w_i \le w_f$ from Lemma 4.12(b) and (c). The last step (12) follows from equation (4).

Case 1.2: The chain C is singleton. We assume that $w_i > R \cdot w_e$, otherwise there is no forward charge from the chain. Then the charge to step t is

$$w_j + w_i - R\cdot w_e \le w_j + w_i - R\cdot w_j \le w_i \le w_f\,,$$

where in the last step we used Lemma 4.12(c).

Case 2: $w_j > w_e$. We claim first that j is not expiring in step t, that is $d_j = t+1$. Indeed, if we had $d_j = t$, then in step $t-1$ the algorithm would have pending packets e and i, plus packet j (pending or lookahead), that need to be scheduled in slots $t-1$ and t. Since $w_e < w_i$ (because step $t-1$ is a chaining step) and $w_e < w_j$ (by the case assumption), packet e could not be in the plan in step $t-1$ which is a contradiction. Thus $d_j = t+1$.

Recall that e is expiring in step $t-1$ by Lemma 4.9(b) and both i and j are considered for the plan in step $t-1$. Moreover, we know that $w_i > w_e$, $w_j > w_e$, $(r_i, d_i) = (t-1, t)$ (by Lemma 4.9(a)), and $(r_j, d_j) = (t, t+1)$. We thus use Lemma 4.2 for step $t-1$ with $a = e, b = i$, and $c = j$, to get that $2\alpha \cdot w_e \ge w_i + w_j$.

Case 2.1: The chain C is long. The charge to step t is

$$\begin{aligned} w_j + (1-\delta)\cdot w_i - (R-1+2\delta)\cdot w_e & \\ &\le w_j + (1-\delta)\cdot w_i - (R-1+2\delta)\cdot \frac{w_i + w_j}{2\alpha} \\ &= \left(1 - \frac{R-1+2\delta}{2\alpha}\right)\cdot w_j + \left(1-\delta-\frac{R-1+2\delta}{2\alpha}\right)\cdot w_i \\ &\le \left(1 - \frac{R-1+2\delta}{2\alpha}\right)\cdot w_f + \left(1-\delta-\frac{R-1+2\delta}{2\alpha}\right)\cdot w_f \qquad (13) \\ &= \left(2 - \delta - \frac{R-1+2\delta}{\alpha}\right)\cdot w_f = R\cdot w_f\,. \qquad (14) \end{aligned}$$

To justify inequality (13), we note that $1-\delta-(R-1+2\delta)/(2\alpha) \ge 0$, by (6), so we can again apply inequalities $w_j \le w_f$ and $w_i \le w_f$ from Lemma 4.12(b) and (c). In the last step (14) we used equation (1).

Case 2.2: The chain C is singleton. We assume that $w_i > R \cdot w_e$, otherwise there is no forward charge from the chain. Then the charge to step t is

$$\begin{aligned} w_j + w_i - R \cdot w_e &\le w_j + w_i - R \cdot \frac{w_i + w_j}{2\alpha} \\ &= \left(1 - \frac{R}{2\alpha}\right) \cdot (w_i + w_j) \\ &\le \left(1 - \frac{R}{2\alpha}\right) \cdot (2w_f) \qquad (15) \\ &= \left(2 - \frac{R}{\alpha}\right) \cdot w_f < R \cdot w_f \,. \qquad (16) \end{aligned}$$

Inequality (15) is valid, because $w_i \le w_f$ and $w_j \le w_f$, by Lemma 4.12, and $1 - R/(2\alpha) \ge 0$ by (7). In step (16) we used (9). □

We now analyze how the forward charge from a chain combines with split charges. First we observe that only the first step from a split-charge pair may receive a forward charge from a chain.

Lemma 4.14. *If j is charged using a split charge to a pair of steps t and t' (where t' is $t+1$ or $t+2$), then t' does not receive a forward charge from a chain.*

Proof. By Lemma 4.4(b) we have $k = f$, which implies that steps t and $t+1$ are not chaining steps. □

Lemma 4.15. *If j is charged using a split charge to a pair of steps t and t', f' is the packet scheduled in t' in ALG, and step t receives a forward charge from a chain C, then the total charge to steps t and t' is at most $R \cdot (w_f + w_{f'})$.*

Proof. First we note that j is expiring in step t by Lemma 4.4(c). Furthermore, i is expiring in step t by Lemma 4.9(a) and f is not expiring in step t by Lemma 4.4(b), so $f \ne i$.

We claim $w_j < w_e$. Indeed, if $w_j > w_e$, then in step $t-1$ the algorithm would have pending packets e and i, plus packet j (pending or lookahead), that need to be scheduled in slots $t-1$ and t. Since $w_e < w_i$ (because step $t-1$ is a chaining step) and $w_e < w_j$, packet e could not be in the plan in step $t-1$ which is a contradiction. Therefore $w_j < w_e$.

Let p_1, p_2, p_3 be the plan at time t. We split the proof into two cases, both having two subcases, one for long chains and one for singleton chains.

Case 1: j is charged using a distant split charge or f' gets a full back charge.

We claim that $2\alpha \cdot w_i < w_f + w_{f'}$. Indeed, since i is expiring and pending in step t by Lemma 4.9(a), we have $w_i \le w_{p_1}$. As the algorithm scheduled $f = p_2$ by Lemma 4.4(d), we get $2\alpha \cdot w_{p_1} < w_f + w_{p_3}$. To prove the claim, it remains to show $w_{f'} \ge w_{p_3}$.

If j is charged using a distant split charge, then by Lemma 4.5 we have $w_g < w_{p_3}$ and in particular, $g \ne p_3$. Thus p_3 is pending and expiring in step $t+2$, hence $w_{p_3} \le w_{f'}$. Otherwise, if j is charged using a close split charge, then $f' = g$ gets a full back charge. Hence $d_{f'} = t+2$. Since also $d_{p_3} = t+2$ and the algorithm chooses the heaviest such packet, we have $w_{p_3} \le w_{f'}$.

The claim follows, since

$$2\alpha \cdot w_i \le 2\alpha \cdot w_{p_1} < w_f + w_{p_3} \le w_f + w_{f'} \,. \qquad (17)$$

Case 1.1: The chain C is long. The total charge to steps t and t', consisting of the full charge to f, a possible full charge to f', the split charge, and the forward charge from the chain, is at most

$$\begin{aligned} w_f + w_{f'} + w_j + (1-\delta)\cdot w_i - (R-1+2\delta)\cdot w_e & \\ &\leq w_f + w_{f'} + (2-R-2\delta)\cdot w_e + (1-\delta)\cdot w_i \\ &< w_f + w_{f'} + (2-R-3\delta)\cdot w_i + w_i && (18) \\ &= w_f + w_{f'} + w_i && (19) \\ &< w_f + w_{f'} + \frac{w_f + w_{f'}}{2\alpha} && (20) \\ &= R\cdot(w_f + w_{f'})\,. \end{aligned}$$

We can use $w_e < w_i$ in (18), because $2 - R - 2\delta \geq 0$ by (5). Equality (19) follows from $2 - R - 3\delta = 0$ by (4) and inequality (20) from (17). In the last step we use (3).

Case 1.2: The chain C is singleton. We suppose that $w_i > R\cdot w_e$, otherwise there is no forward charge from the chain. We upper bound the total charge to steps t and t' by

$$\begin{aligned} w_f + w_{f'} + w_j + w_i - R\cdot w_e &\leq w_f + w_{f'} + (1-R)\cdot w_e + w_i \\ &< w_f + w_{f'} + w_i \\ &< w_f + w_{f'} + \frac{w_f + w_{f'}}{2\alpha} && (21) \\ &= R\cdot(w_f + w_{f'})\,, \end{aligned}$$

where we apply Equation 17 in (21), and we use (3) in the last step.

Case 2: j is charged using a close split charge and $f' = g$ does not get a full back charge. Observe that in this case g does not receive any full charge as $k = f$ is charged by a full back charge. We have (i) $2\alpha\cdot w_{p_1} < w_f + w_g$, or (ii) $2\alpha\cdot(w_{p_1} - w_g) < w_f + w_g$ by the definition of the close split charge. We suppose that we have (ii), since (i) is stronger than (ii).

Since i is expiring and pending in step t by Lemma 4.9(a), we have $w_i \leq w_{p_1}$. Hence $2\alpha\cdot(w_i - w_g) < w_f + w_g$. This is equivalent to

$$w_i < \frac{w_f + (2\alpha+1)\cdot w_g}{2\alpha}\,. \tag{22}$$

Case 2.1: The chain C is long. The total charge to steps t and $t' = t+1$ is

$$\begin{aligned} w_f + w_j + (1-\delta)\cdot w_i - (R-1+2\delta)\cdot w_e & \\ &\leq w_f + (2-R-2\delta)\cdot w_e + (1-\delta)\cdot w_i \\ &< w_f + (2-R-3\delta)\cdot w_i + w_i && (23) \\ &= w_f + w_i && (24) \\ &< w_f + \frac{w_f + (2\alpha+1)\cdot w_{f'}}{2\alpha} && (25) \\ &= w_f + w_{f'} + \frac{w_f + w_{f'}}{2\alpha} \\ &= R\cdot(w_f + w_{f'})\,, \end{aligned}$$

We can use $w_e < w_i$ in (23), because $2 - R - 2\delta \geq 0$ by (5). Then we use $2 - R - 3\delta = 0$ by (4) in (24), Equation 22 in (25), and Equation 3 in the last step.

Case 2.2: The chain C is singleton. We again suppose that $w_i > R \cdot w_e$, as otherwise there is no forward charge from the chain. We upper bound the total charge to steps t and $t+1$ by

$$\begin{aligned} w_f + w_j + w_i - R \cdot w_e &\leq w_f + (1-R) \cdot w_e + w_i \\ &< w_f + w_i \\ &< w_f + \frac{w_f + (2\alpha+1) \cdot w_{f'}}{2\alpha} \qquad (26) \\ &= w_f + w_{f'} + \frac{w_f + w_{f'}}{2\alpha} \\ &= R \cdot (w_f + w_{f'}) \,, \end{aligned}$$

where we apply (22) in inequality (26), and (3) in the last step. □

We now summarize our analysis of CompareWithBias(α). If t is not in a split-charge pair, we show upper bounds on the total charge to step t. For each split-charge pair (t, t'), we show upper bounds on the total charge to both steps t and t'. This is sufficient, since split-charge pairs are pairwise disjoint by Lemma 4.7, thus summing all the bounds gives the result in Theorem 4.1.

For each step t, we distinguish three cases according to whether t is in a split-charge pair and whether t is a chaining step. In all cases, let f be the packet scheduled at time t in ALG and let j be the packet scheduled at time t in OPT.

Case 1: Step t is not chaining and it is not in a split-charge pair. Then t receives at most one full charge from a packet p such that $w_p \leq w_f$ (by Lemma 4.3 and charging rules) and possibly a forward charge from a chain C; then Lemma 4.13 shows that the sum of a forward charge from a chain and a full charge is at most $R \cdot w_f$.

Case 2: Step t is a chaining step. Then it does not receive a split charge or a full charge, by Lemma 4.10. Lemma 4.11 implies that step t receives a charge of at most $R \cdot w_f$.

Case 3: (t, t') is a split-charge pair, i.e., t is the first step of the split-charge pair and $t' = t+1$, or $t' = t+2$. Thus j is charged using a split charge. Let f' be the packet scheduled in step t' in ALG.

By Lemma 4.14 step t' does not receive a forward charge from a chain. If step t also does not receive a forward charge from a chain, then the total charge to steps t and t' is at most $R \cdot (w_f + w_{f'})$ by Lemma 4.8. Otherwise, step t receives a forward charge from a chain and we apply Lemma 4.15 to show that the total charge to steps t and t' is again at most $R \cdot (w_f + w_{f'})$.

5 A Lower Bound for 2-bounded Instances with Lookahead

In this section, we prove that there is no deterministic online algorithm for PacketScheduling with ℓ-lookahead that has competitive ratio smaller than $R := \frac{1}{2(\ell+1)}(1 + \sqrt{5 + 8\ell + 4\ell^2})$ for any $\ell \geq 0$, even for 2-bounded instances. We note that $R > 1$ for any $\ell \geq 0$, that R tends to 1 as ℓ goes to infinity, and that R is the positive root of the quadratic equation

$$(\ell+1)R^2 - R - (\ell+1) = 0 \,. \qquad (27)$$

The idea of our proof is similar to the proof of the lower bound of ϕ for PacketScheduling [11, 2, 6] and, indeed, for $\ell = 0$ our lower bound is equal to ϕ. For the case of 1-lookahead we obtain a lower bound of $\frac{1}{4}(1 + \sqrt{17}) \approx 1.281$.

Theorem 5.1. *Let $\ell \geq 0$ be an integer and $R = \frac{1}{2(\ell+1)}(1 + \sqrt{5 + 8\ell + 4\ell^2})$. For each $\varepsilon > 0$, no deterministic online algorithm for PacketScheduling with ℓ-lookahead can be $(R - \varepsilon)$-competitive, even for 2-bounded instances.*

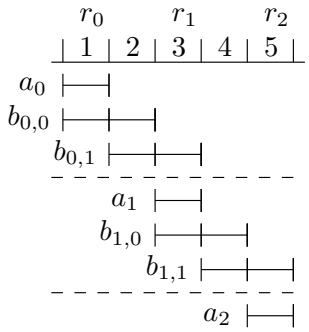

Figure 5: The instance for $\ell = 1$ and $k = n = 2$. Each packet has a row which shows slots between its release time and deadline. Packets from different phases are separated by a dashed line.

Proof. Fix some online algorithm $\mathcal{A}$ and some $\varepsilon > 0$. We will show that, for some sufficiently large integer n and sufficiently small $\delta > 0$, there is a 2-bounded instance of PacketScheduling with ℓ-lookahead, parameterized by n and δ, for which the optimal profit is at least $(R - \varepsilon)$ times the profit of $\mathcal{A}$.

Our instance will consist of phases $0, \ldots, k$, for some $k \leq n$. The number k of phases is determined by the adversary based on the behavior of $\mathcal{A}$. Each phase (except phase n) will involve $\ell + 2$ packets. The weights of these packets will grow roughly exponentially from one phase to next.

The adversary strategy is as follows. We start with phase 0. Suppose that some phase i, where $0 \leq i < n$, has been reached. Let $r_i = (\ell + 1)i + 1$ be the first slot of phase i. In phase i the adversary releases the following $\ell + 2$ packets:

- A packet a_i with weight w_i, release time r_i and deadline r_i, i.e., a tight packet.
- Packets $b_{i,j}$ for $j = 0, \ldots, \ell$ with weight w_{i+1}, release time $r_i + j$ and deadline $r_i + j + 1$.

(The weights w_i will be specified later.) Now, if $\mathcal{A}$ schedules an expiring packet in step r_i (a tight packet a_i or $b_{i-1,\ell}$, which may be pending from the previous phase), then the game continues; the adversary will proceed to phase $i + 1$. Otherwise, the algorithm schedules packet $b_{i,0}$, in which case the adversary lets $k = i$ and the game ends. Note that in steps $r_i + 1, \ldots, r_i + \ell$ the algorithm may schedule only $b_{i,j}$ (for some j) of weight w_{i+1}. Also, importantly, in step r_i the algorithm cannot yet see whether the packets from phase $i + 1$ will arrive or not.

If phase $i = n$ is reached, then $k = n$, and in phase n the adversary releases a single tight packet a_n with weight w_n and release time and deadline r_n. See Figure 5 for an illustration.

We calculate the ratio between the weight of packets in an optimal schedule and the weight of packets sent by the algorithm. Let $S_k = \sum_{i=0}^{k} w_i$. There are two cases: either $k < n$, or $k = n$.

Case 1: $k < n$. In all steps r_i for $i < k$ algorithm $\mathcal{A}$ scheduled an expiring packet of weight w_i and in step r_k it scheduled packet $b_{k,0}$ of weight w_{k+1}. In steps $r_i + 1, \ldots, r_i + \ell$ for $i < k$ it scheduled packets of weight w_{i+1}. Finally, in phase k the algorithm scheduled $\ell + 1$ packets of weight w_{k+1}, including $b_{k,0}$. Overall, $\mathcal{A}$ scheduled packets of total weight $S_{k-1} + \ell \cdot (S_k - w_0) + (\ell + 1) \cdot w_{k+1} = (\ell + 1) \cdot S_{k+1} - w_k - \ell \cdot w_0$.

The adversary schedules packets of weight w_{i+1} in steps $r_i, \ldots, r_i + \ell$ for $i < k$ and all packets from phase k in steps $r_k, \ldots r_k + \ell + 1$. In total, the optimum has a schedule of weight $(\ell + 1) \cdot (S_k - w_0) + w_k + (\ell + 1) \cdot w_{k+1} = (\ell + 1) \cdot S_{k+1} + w_k - (\ell + 1) \cdot w_0$. The ratio is

$$R_k = \frac{(\ell + 1) \cdot S_{k+1} + w_k - (\ell + 1) \cdot w_0}{(\ell + 1) \cdot S_{k+1} - w_k - \ell \cdot w_0}.$$

Case 2: $k = n$. As before, in all steps r_i for $i < n$ algorithm $\mathcal{A}$ scheduled an expiring packet of weight w_i and in steps $r_i + 1, \ldots, r_i + \ell$ for $i < n$ it scheduled a packet of weight w_{i+1}. In the last step r_n it scheduled a packet of weight w_n as there is no other choice. Overall, the total weight of the $\mathcal{A}$'s schedule is $S_{n-1} + \ell \cdot (S_n - w_0) + w_n = (\ell + 1) \cdot S_n - \ell \cdot w_0$.

The adversary schedules packets of weight w_{i+1} in steps $r_i, \ldots, r_i + \ell$ for $i < n$ and a packet of weight w_n in the last step r_n which adds up to $(\ell + 1) \cdot S_n + w_n - (\ell + 1) \cdot w_0$. The ratio is

$$\widehat{R}_n = \frac{(\ell + 1) \cdot S_n + w_n - (\ell + 1) \cdot w_0}{(\ell + 1) \cdot S_n - \ell \cdot w_0}.$$

We normalize the instances so that $w_0 = 1$. It remains to show that we can set the weights so that $R_k \geq R - \varepsilon$ for all $k \geq 0$ and $\widehat{R}_n \geq R - \varepsilon$.

We first define a sequence of weights, parametrized by some parameter $\delta \geq 0$, such that $R_k = R$ for all $k \geq 1$. Using $w_k = S_k - S_{k-1}$ for $k \geq 1$ and $w_0 = 1$, the condition $R_k = R$ for $k \geq 1$ is rewritten as

$$R = \frac{(\ell + 1) \cdot S_{k+1} + S_k - S_{k-1} - (\ell + 1)}{(\ell + 1) \cdot S_{k+1} - S_k + S_{k-1} - \ell},$$

or equivalently as

$$(\ell + 1)(R - 1)S_{k+1} - (R + 1)S_k + (R + 1)S_{k-1} = \ell R - (\ell + 1). \tag{28}$$

By (27) we get that $(\ell + 1)(R - 1) = R/(R + 1)$ and similarly $\ell R - (\ell + 1) = -R^2/(R + 1)$. Substituting and multiplying by $R + 1$, we obtain that (28) is equivalent to

$$R \cdot S_{k+1} - (R + 1)^2 S_k + (R + 1)^2 S_{k-1} = -R^2. \tag{29}$$

To define our instance, we set $w_0 = 1$ and for $i = 1, 2, \ldots$,

$$w_i = (\gamma + 1)\alpha^{i-1}(\alpha - 1) + \delta[\beta^{i-1}(\beta - 1) - \alpha^{i-1}(\alpha - 1)],$$

where

$$\alpha = 1 + \frac{1}{R} = \frac{R + 1}{R}, \quad \beta = R + 1, \quad \gamma = R,$$

and $\delta > 0$ is a parameter to be chosen later. Summing the geometric sequences in $S_k = \sum_{i=0}^{k} w_i$, we obtain that, for $k = 0, \ldots, n$,

$$S_k = (\gamma + 1)\alpha^k + \delta(\beta^k - \alpha^k) - \gamma. \tag{30}$$

It can be verified that (29) holds and thus, for any choice of δ and any $k \geq 1$, we have $R_k = R$. In fact, (30) describes a general solution of the linear recurrence (29) that satisfies one initial condition $S_0 = w_0 = 1$, as α, β are the two roots of $Rx^2 - (R + 1)^2 x + (R + 1)^2$ which is the characteristic polynomial of the recurrence, and $S_0 = S_1 = \cdots = S_n = -\gamma$ is a particular solution of the recurrence; furthermore, for $\delta = 0$ (30) gives a particular solution satisfying $S_0 = 1$ and changing δ does not change S_0.

We now show that for $\delta = 0$ the solution would satisfy $R_0 = R$. We first calculate w_1:

$$w_1 = (\gamma + 1)(\alpha - 1) = (R + 1) \cdot \frac{1}{R} = \alpha.$$

By (27) we have $(\ell + 1)\alpha = 1/(R - 1)$. Using this we can calculate R_0 as

$$R_0 = \frac{(\ell + 1)w_1 + 1}{(\ell + 1)w_1} = 1 + \frac{1}{(\ell + 1)\alpha} = R.$$

By continuity of the dependence of w_1 and R_0 on δ, for a sufficiently small $\delta > 0$ we have $R_0 \geq R - \varepsilon$. We fix such a $\delta > 0$.

Since $1 < \alpha < \beta$, for $n \to \infty$, the dominating term in S_n is $\delta\beta^n$ and w_0 is negligible compared to S_n. Thus we obtain

$$\lim_{n\to\infty} \widehat{R}_n = \lim_{n\to\infty} \frac{(\ell+1)S_n + S_n - S_{n-1}}{(\ell+1)S_n} = \lim_{n\to\infty} \frac{(\ell+2)\delta\beta^n - \delta\beta^{n-1}}{(\ell+1)\delta\beta^n} = \frac{(\ell+2)\beta - 1}{(\ell+1)\beta} = R\,.$$

The last equality follows from (27). Actually, this is the equation that defines R as the optimal ratio for our construction (if β is expressed in terms of R as the root of the characteristic polynomial). Consequently, for some sufficiently large n, we have that $\widehat{R}_n \geq R - \varepsilon$. Fix this n.

Summarizing, we showed that for any $\varepsilon > 0$ the adversary can choose $\delta > 0$ and n, and an instance with up to n phases, such that for this instance we have $R_0 \geq R - \varepsilon$, $R_k = R$ for all $k \geq 1$, and $\widehat{R}_n \geq R - \varepsilon$. This implies that the competitive ratio of $\mathcal{A}$ is at least R, completing the proof. □

6 A Lower Bound for Radomized Algorithms with Lookahead

In this section we consider arbitrary instances — without the s-bounded restriction. We show that, somewhat counter-intuitively, it is not possible to achieve a competitive ratio arbitrarily close to 1 by using sufficiently large lookahead. In fact, it is not possible to achieve ratio lower than 1.25 even by a randomized algorithm against the oblivious adversary. Our argument is a generalization of the one by Chin and Fung [6] for 2-bounded instances without lookahead.

Theorem 6.1. *For any fixed $\ell \geq 0$, there is no better than* 1.25*-competitive randomized algorithm with ℓ-lookahead.*

Proof. We use the easy direction of the Yao's minimax principle for online algorithms (see e.g. [4]). This principle states that the competitive ratio of a randomized algorithm on the worst-case instance can be lower bounded by the performance of deterministic algorithms with inputs drawn from a suitable probability distribution of instances. More precisely, suppose that for each deterministic algorithm, its expected profit on an input drawn from this distribution is at most $1/R$ of the expected optimal profit for the same distribution of the inputs. Then the competitive ratio of every randomized algorithm is at least R.

Let $n, k \gg \ell$ be large integers. We define a set of $n+1$ instances as follows (each instance is a set of packets, with each packet p specified by a triple (r_p, d_p, w_p)):

$$\begin{aligned}
J_0 &= \{a_{0,m} = (1, k, 1), b_{0,m} = (1, 2k, 2) \,|\, m = 1, \ldots, k\},\\
J_i &= J_{i-1} \cup \{a_{i,m} = (ik+1, (i+1)k, 2^i), b_{i,m} = (ik+1, (i+2)k, 2^{i+1}) \,|\, m = 1, \ldots, k\}\\
&\qquad \text{for } i = 1, \ldots, n-1,\\
J_n &= J_{n-1} \cup \{a_{n,m} = (nk+1, (n+1)k, 2^n) \,|\, m = 1, \ldots, k\}\,.
\end{aligned}$$

A *span* of a packet (r_p, d_p, w_p) is defined as $d_p - r_p + 1$. Note that this equals the number of slots where the packet can be scheduled, corresponding to the informal definition of span in the introduction. Then, in words, in instance J_i, $i = 0, \ldots, n-1$, in step $ik+1$ the adversary releases k packets $a_{i,1}, \ldots, a_{i,k}$ of span k and k packets $b_{i,1}, \ldots, b_{i,k}$ of span $2k$ in addition to packets from instance J_{i-1}. In instance J_n it additionally releases k packets $a_{n,1}, \ldots, a_{n,k}$ of span k only. See Figure 6 for an example. For $i = 0, \ldots, n$, we call the interval $[ik+1, (i+1)k]$ of slots the i-th *phase*. Note that in each phase except the last one, pending packets are of two types: lighter packets expiring in that phase and heavier packets expiring in the next phase.

We define the probability distribution of the instances J_i such that for $i = 0, \ldots, n-1$, instance J_i is drawn with probability $p_i = 1/2^{i+1}$ and instance J_n is drawn with probability $p_n = 1/2^n$. Clearly, $\sum_i p_i = 1$.

First, we analyze the offline optimum profits. Observe that on instance J_i, $i = 1, \ldots, n-1$, in the j-th phase for $j = 0, \ldots, i-1$ the adversary schedules k heavier packets (of span $2k$), which are expiring in the next phase, while in each of the last two phases it transmits k packets expiring in the phase. Thus its profit is $\mathsf{OPT}(J_i) = \sum_{j=0}^{i-1} k \cdot 2^{j+1} + k \cdot 2^i + k \cdot 2^{i+1} = k \cdot (2^{i+2} + 2^i - 2)$. Similarly, it holds that $\mathsf{OPT}(J_n) = \sum_{j=0}^{n-1} k \cdot 2^{j+1} + k \cdot 2^n = k \cdot (2^{n+1} + 2^n - 2)$. It follows that the expected optimum profit is

$$\begin{aligned}
\mathsf{E}\left[\mathsf{OPT}\right] &= \sum_{i=0}^{n} p_i \cdot \mathsf{OPT}(J_i) = \sum_{i=0}^{n-1} \frac{k \cdot (2^{i+2} + 2^i - 2)}{2^{i+1}} + \frac{k \cdot (2^{n+1} + 2^n - 2)}{2^n} \\
&= k \cdot \sum_{i=0}^{n-1} \left(2 + \frac{1}{2} - \frac{1}{2^i}\right) + k \cdot \left(2 + 1 - \frac{1}{2^{n-1}}\right) \\
&= k \cdot \left(\frac{5}{2}n - 2 + \frac{1}{2^{n-1}} + 3 - \frac{1}{2^{n-1}}\right) = k \cdot \left(\frac{5}{2}n + 1\right) .
\end{aligned}$$

Fix a deterministic algorithm ALG. To upper bound the expected profit of ALG, we first get rid of the influence of lookahead, which affects its behavior only in the last ℓ steps in each phase. Namely, we make the following assumption about the behavior of ALG on instance J_i without loss of generality: For each phase $j = 0, \ldots, i-1$, in the last ℓ slots of the phase (when ALG knows that more packets will arrive) ALG schedules packets not expiring in the j-th phase only, i.e., the heaviest pending packets. Moreover, if $i < n$, in the last ℓ slots of the i-th phase (which is the penultimate phase) it sends packets expiring in that phase only.

Let $\alpha_i, i = 0, \ldots, n-1$, be the number of lighter packets that the algorithm schedules in the first $k - \ell$ slots of the phase. We get the following expressions of the algorithm's profits on the instances:

$$\begin{aligned}
\mathsf{ALG}(J_i) &= \sum_{j=0}^{i-1} \left(\alpha_j \cdot 2^j + (k - \ell - \alpha_j + \ell) \cdot 2^{j+1}\right) + (\alpha_i + \ell) \cdot 2^i + k \cdot 2^{i+1} \\
&= \sum_{j=0}^{i-1} \left(k \cdot 2^{j+1} - \alpha_j \cdot 2^j\right) + \alpha_i \cdot 2^i + \ell \cdot 2^i + k \cdot 2^{i+1} \\
&\leq k \cdot 2^{i+2} + \ell \cdot 2^i - \sum_{j=0}^{i-1} \alpha_j \cdot 2^j + \alpha_i \cdot 2^i \\
\mathsf{ALG}(J_n) &= \sum_{j=0}^{n-1} \left(\alpha_j \cdot 2^j + (k - \ell - \alpha_j + \ell) \cdot 2^{j+1}\right) + k \cdot 2^n \\
&= \sum_{j=0}^{n-1} \left(k \cdot 2^{j+1} - \alpha_j \cdot 2^j\right) + k \cdot 2^n \\
&\leq k \cdot 2^{n+1} + k \cdot 2^n - \sum_{j=0}^{n-1} \alpha_j \cdot 2^j .
\end{aligned}$$

Our goal is to express $\mathsf{E}\left[\mathsf{ALG}\right] = \sum_i p_i \cdot \mathsf{ALG}(J_i)$. We now show that for any $j = 0, \ldots, n-1$ the coefficient of α_j in $\mathsf{E}\left[\mathsf{ALG}\right]$ is 0 by simply summing the coefficients of α_j over all i in the bounds on $\mathsf{ALG}(J_i)$ multiplied by p_i. We note that α_j appears with positive coefficient 2^j in

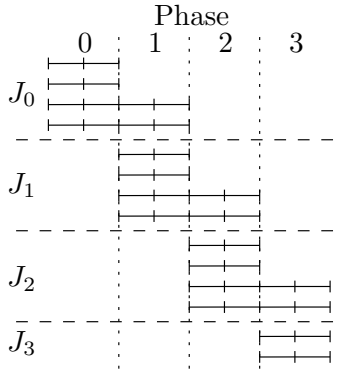

Figure 6: An illustration of the lower bound of 1.25 with $k = 2$ and $n = 3$. The dotted vertical lines split slots into phases.

the bound for $\mathsf{ALG}(J_j)$ and with negative coefficients -2^j in the bounds for $\mathsf{ALG}(J_i)$ for all $i = j+1, \ldots, n-1, n$. We thus obtain that the sum is

$$\frac{2^j}{2^{j+1}} - \sum_{i=j+1}^{n-1} \frac{2^j}{2^{i+1}} - \frac{2^j}{2^n} = 0$$

It follows that

$$\begin{aligned}\mathsf{E}\left[\mathsf{ALG}\right] &\leq \sum_{i=0}^{n-1} \frac{k \cdot 2^{i+2} + \ell 2^i}{2^{i+1}} + \frac{k \cdot 2^{n+1} + k \cdot 2^n}{2^n} \\ &= k \cdot n \cdot 2 + \ell \cdot n \cdot \frac{1}{2} + k \cdot 3 = k \cdot (2n+3) + \ell \cdot \frac{1}{2} n\,.\end{aligned}$$

We conclude that the ratio $\mathsf{E}\left[\mathsf{OPT}\right] / \mathsf{E}\left[\mathsf{ALG}\right]$ tends to 1.25 if n and k go to infinity. □

Note that in the case $\ell = 0$ (i.e., no lookahead), we can choose $k = 1$ and then the construction is the same as in [6].

References

[1] Susanne Albers. On the influence of lookahead in competitive paging algorithms. *Algorithmica*, 18(3):283–305, 1997.

[2] Nir Andelman, Yishay Mansour, and An Zhu. Competitive queueing policies for QoS switches. In *Proc. 14th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'03)*, pages 761–770, 2003.

[3] Martin Böhm, Marek Chrobak, Lukasz Jez, Fei Li, Jirí Sgall, and Pavel Veselý. Online packet scheduling with bounded delay and lookahead. In *Proc. 27th International Symposium on Algorithms and Computation (ISAAC'16)*, pages 21:1–21:13, 2016.

[4] Allan Borodin and Ran El-Yaniv. *Online computation and competitive analysis*. Cambridge University Press, 1998.

[5] Francis Y. L. Chin, Marek Chrobak, Stanley P. Y. Fung, Wojciech Jawor, Jiří Sgall, and Tomáš Tichý. Online competitive algorithms for maximizing weighted throughput of unit jobs. *J. of Discrete Algorithms*, 4(2):255–276, 2006.

[6] Francis Y. L. Chin and Stanley P. Y. Fung. Online scheduling with partial job values: Does timesharing or randomization help? *Algorithmica*, 37(3):149–164, 2003.

[7] Marek Chrobak, Wojciech Jawor, Jiří Sgall, and Tomáš Tichý. Improved online algorithms for buffer management in QoS switches. *ACM Trans. Algorithms*, 3(4), 2007.

[8] Matthias Englert and Matthias Westermann. Considering suppressed packets improves buffer management in quality of service switches. *SIAM Journal on Computing*, 41(5):1166–1192, 2012.

[9] Michael H. Goldwasser. A survey of buffer management policies for packet switches. *SIGACT News*, 41(1):100–128, 2010.

[10] Edward F. Grove. Online bin packing with lookahead. In *Proc. 6th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'95)*, pages 430–436, 1995.

[11] Bruce Hajek. On the competitiveness of on-line scheduling of unit-length packets with hard deadlines in slotted time. In *Proc. Conference on Information Sciences and Systems*, pages 434–438, 2001.

[12] Alexander Kesselman, Zvi Lotker, Yishay Mansour, Boaz Patt-Shamir, Baruch Schieber, and Maxim Sviridenko. Buffer overflow management in QoS switches. *SIAM Journal on Computing*, 33(3):563–583, 2004.

[13] Koji M Kobayashi. An optimal algorithm for 2-bounded delay buffer management with lookahead. *arXiv preprint arXiv:1807.00121*, June 2018.

[14] Fei Li, Jay Sethuraman, and Clifford Stein. An optimal online algorithm for packet scheduling with agreeable deadlines. In *Proc. 16th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'05)*, pages 801–802, 2005.

[15] Rajeev Motwani, Vijay Saraswat, and Eric Torng. Online scheduling with lookahead: Multipass assembly lines. *INFORMS J. on Computing*, 10(3):331–340, 1998.

[16] Pavel Veselý, Marek Chrobak, Łukasz Jeż, and Jiří Sgall. A ϕ-competitive algorithm for scheduling packets with deadlines. *arXiv preprint arXiv:1807.07177*, July 2018.